A Model for Composite System Design

E. Doerry, S. Fickas, R. Helm

Department of Computer Science
University of Oregon
Eugene, OR 97403
USA

M. Feather

USC / Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
USA

1 Our goal - a model for composite system design

Composite systems are systems that encompass multiple agents involved in ongoing, interactive activities. We wish to study requirements acquisition, specification and design of such systems.

The benefit of this approach (and what distinguishes it from previous work) is that it makes the interaction/interface between agents a first-class concern, to be explicitly represented and reasoned about. Thus, design problems concerned with a number of cooperating/interacting agents are particularly well-suited for treatment as composite systems, while those in which agents are fairly independent could just as well be handled by other approaches. Another benefit is that this approach is conducive towards incorporating domain-specific knowledge into the design process.

We center our efforts around a style of specification which comprises a description of *possible* system behaviors, together with a set of 'goals', namely constraints on those possible behaviors. Together these denote those and only those behaviors that satisfy all the constraints. Since we are concerned with composite systems, initial expressions of behaviors and goals will typically involve multiple agents. In these cases it is a challenging design activity to decompose these descriptions into behaviors of the individual agents, such that the composition of those agent behaviors will achieve the desired system behaviors. Communication between agents will typically be required to permit the coordination and transfer of information necessary to achieve goals, and derivation of this communication is a significant part of the design activity. Furthermore, we wish to encompass the notions of reliability and motivation — important in real-world systems where perfect behavior of components (mechanical or human) cannot be assumed. Finally, we believe any formal model of specification must include non-functional requirements such as cost, safety, and human factors.

Our research objective is a model that encompasses this entire design activity, and thereafter, techniques and tools called for by the model, which will serve to provide automated assistance to a skilled designer of composite systems. Herein we outline the proposed model that we have established so far, together with the experimental pieces of technology that we are assembling. Since this is ongoing research, we may expect that our model will undergo further refinement, and our selection of tools may require reconsideration. For these reasons we have chosen to present this as a position statement rather than as a paper of partial results.

2 The Model

2.1 Concepts within the model

Specifications denote behaviors, that is, sequences of activities. As a running example, we consider the specification of elevators[1] serving passengers in a multi-story building; this task is simple but intuitive, and so serves as a convenient illustration of our concepts. In this context, behaviors involve activities of passengers entering elevators, elevator doors closing, elevators moving, etc. Specifications comprise two parts:

- a *'generative'* part denoting, for each agent, the capabilities of that agent, e.g., passenger - enter or exit elevator; elevator - move up or down, and
- a *'constraining'* part consisting of a set of 'goals', i.e., constraints on behavior. For the purpose of

[1] 'Lifts', in British terminology.

TH0388-9/91/0000/0216$01.00 © 1991 IEEE

describing composite systems, it is often convenient to express goals in terms of system-wide properties, regardless of how that system is decomposed into agents. For example, passengers shall never be moved further from their destinations. The end point of our design activity may still be a specification insofar as it is free of traditional algorithmic concerns, for example, but must make no use of system-wide expressions of properties.

During the course of development, goals are replaced with 'responsibilities' assigned to subsets of agents. Informally, only those agents responsible for a goal are expected to limit their own behavior to ensure satisfaction of that goal. For example, if the elevator system alone is responsible for keeping passengers from falling down elevator shafts, then the system must keep doors closed when necessary rather than rely upon passengers to limit their choice of when to walk through an open doorway. Ultimately it is necessary to subdivide responsibility so as to be able to assign individual pieces to individual agents.

We also reason about what information is available to what agents. For example, passengers know what their own destinations are, but not, generally, the destinations of other passengers. The set of agents (possibly a single agent) with some responsibility must have sufficient information to be able to correctly select from among their possible activities. For example, the elevator system must know which way to move an elevator so as to transport a passenger inside that elevator toward his or her destination. If information is required but not available, we must modify the specification to provide that information, or modify the responsibilities and/or their assignments so as to resolve this inconsistency. The particular information that a set of agents needs from the rest of the system is expressed as an 'interface requirement'. For example, the elevator system's need to know in which direction a waiting passenger wishes to be transported. For an interface requirement, we design an 'interface specification', namely some particular way of providing that information. This takes the form of an extension of the specification of agent capabilities. For example, a direction input device, which new passengers at a floor are expected to use to summon an elevator.

Reliability and motivation are superimposed on the notions of agent capabilities and choice. Conventional notions of reliability (e.g., probability of failure of some mechanism) can be attached to activities, including those introduced to satisfy interface requirements. For example, the indicator of elevator direction might fail. Furthermore, reliability can be associated with making choices, so that we can reason about the likelihood and effect of an agent making a 'wrong' choice by mistake. For example, a passenger may occasionally press the wrong direction button when summoning an elevator. Motivation applies primarily to rational agents, who may be expected to have objectives of their own. An agent's personal objectives and the agent's responsibilities, derived from system-wide goals, will be either *reinforcing*, *neutral*, or *conflicting*. Neutral and conflicting cases will require further design activity to motivate the agent to choose so as to further the system-wide goals, or to modify the design so as to be able to recover from detrimental choices.

2.2 Activities within the model

We have experimented with the design of several composite systems using various formalisms. We expect both the form of our representation and the design process model we use will change in these early stages. However, we have identified a set of activities we feel are endemic to composite system design (CSD) in general:

1. Identifying problematic behaviors. We rely on a scenario generator [2, 1] to find counterexamples, i.e., behaviors that demonstrate a goal is not being met by the current model of agent cooperation. These counterexamples have two roles: 1) they let us know if a goal is achieved in the current model, and if not, 2) they act as a guide on how the goal or model can be changed to bring achievement [3].

2. Modifying goals. There is nothing held inviolate to change in our approach. Thus, goals themselves may be modified. For instance, if no cost effective design can be found, we may need to weaken/approximate a goal. While we currently lack a formal theory of goal modification in CSD, it appears from our example designs that a tractable set of goal transformations exist. This is an area of active interest to us [8].

3. Assigning responsibility. This is perhaps the key step in our approach. There are two questions of interest: 1) what are the semantics of responsibility assignment [5, 4], and 2) what criteria are available for choosing among alternative assignments [6]. For the second question, in particular, major design decisions are made at this step. It is here where a large analysis effort, e.g., critics, comparators, cost models, safety models, will pay off. In some areas of CSD (e.g., many engineering fields), formal evaluation models exist. Our interest in these cases is integrating cost/safety tables, equations, etc., into our design

model. In other CSD areas (e.g., Human-Computer Interface design), no formal evaluation models exist. Our interest in these cases is with approximation techniques. For instance, we have looked at Qualitative Reasoning [3] as a means of approximating the degree of cost of certain design decisions.

4. Deriving an interface requirement. When an agent is assigned the responsibility of achieving a goal, it must limit its actions to achieve that goal. If it can do so solely with local information, then no new interface component is needed. More typically, the agent will require non-local information, i.e., information from other agents. The information needed defines an interface requirement for the agent: Thus, if one were to ask why *the need* for the floor buttons inside an elevator, we would answer that it was a requirement of the elevator to gain this information to limit its movement to the destination floor.

5. Deriving an interface specification. Deciding how an interface requirement will be met is another key design decision. Among other things, we must reason about the cost and availability of information within our model of the world. The issues of evaluating alternatives here is similar to that of responsibility assignment (and in fact, can be viewed as a type of responsibility issue: given a goal of obtaining certain information, what agents will act to provide the necessary information). Returning to the floor buttons example, if one were to ask why the need for the *floor buttons*, we would answer that the designer specified that the user would supply the required information through a type of input device. This is the interface specification. The actual choice of floor buttons over, say, voice input is an implementation decision that our design model stops short of.

6. Reasoning about motivation and reliability. While both reliability and motivation can be viewed as a type of evaluation criteria, and hence fall under responsibility assignment or interface specification, we have separated them here because of the special interest they hold for us. The need for each derived from our efforts to reverse engineer existing CSDs using our design model. We found that without the reliability and motivation criteria, it was impossible for us to rationalize the design of these existing composite systems. Looking at elevator interfaces, for example, bells, lights, and even doors are justified, at least in part, by human reliability issues. Some elevator systems make use of weight sensors that can detect when the elevator is empty, using this information to know when to cancel floor requests internal to the elevator when there is no-one left inside (such requests might stem from mistakes — "oops, I pressed the 12th floor button instead of the 10th", dynamic revision of destination — "ok, I'll get off at this floor to go directly to the meeting with you", or pranksters — "let's press all the floor buttons as we get out"); some elevators have vandalism detectors to sound alarms. Their presence can only by justified by looking at irresponsible agent behavior and attempts to minimize it or recover from it. Irresponsible behavior falls under what we consider motivation issues. Some areas of CSD address reliability issues of physical and human agents formally. As with cost or safety criteria, we would like to find a means of incorporating existing reliability tables and equations into our approach. Unfortunately, there is no field that has produced useful formal models of motivation as it applies to CSD. Worse, it is unclear that we can ever have a satisfactory formal behavior model that would allow us to predict the need for weight sensors or vandalism alarms. In practice, such devices evolve from experience in elevator design. Capturing such experience does seem a more tractable problem, and is one we have started exploring through Case-Based Reasoning techniques [7].

Acknowledgements

We especially thank the following people for their insights during our discussions on composite system design and related issues: co-organizers of the 1991 AAAI Spring Symposium on Composite System Design, Les Gasser and Lewis Johnson; other members of the Kate group at the University of Oregon, Brian Durney, Bill Robinson and John Anderson. Feather has been supported in part by Defense Advanced Research Projects Agency grant No. NCC-2-520, and in part by Rome Air Development Center contract No. F30602-89-C-0103. All four authors have been supported by NSF grant No. CCR-8804085. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official opinion or policy of DARPA, RADC, NSF, the U.S. Government, or any other person or agency connected with them.

References

[1] J. Anderson and S. Fickas. A proposed perspective shift: viewing specification design as a planning problem. In *Proceedings, 5th International Workshop on Software Specification and Design, Pittsburgh, Pennsylvania, USA*, pages 177–184. Computer Society Press of the IEEE, 1989.

[2] K. Benner. Simulation in support of specification validation. In *Proceedings of the 5th Annual RADC Knowledge-Based Software Assistant (KBSA) Conference, Liverpool, NY, September 1990*, pages 305–316, 1990.

[3] K. Downing and S. Fickas. Specification criticism via goal-directed envisionment. In *Proceedings, 6th International Workshop on Software Specification and Design, Lake Como, Italy*, 1991.

[4] E. Dubois. A logic of action for supporting goal-oriented elaborations of requirements. In *Proceedings, 5th International Workshop on Software Specification and Design, Pittsburgh, Pennsylvania, USA*, pages 160–168. Computer Society Press of the IEEE, 1989.

[5] M.S. Feather. Language support for the specification and development of composite systems. *ACM Transactions on Programming Languages and Systems*, 9(2):198–234, April 1987.

[6] S. Fickas and R. Helm. A transformational approach to composite system specification. Technical Report TR-90-19, CS Dept., U of Oregon, 1990.

[7] S. Fickas and P. Nagarajan. Being suspicious: critiquing problem specifications. In *Proceedings, AAAI Conference, Minneapolis*, 1988.

[8] B. Robinson. A multi-agent view of requirements. In *Proceedings of the International Conference on Software Engineering, Nice*, 1990.